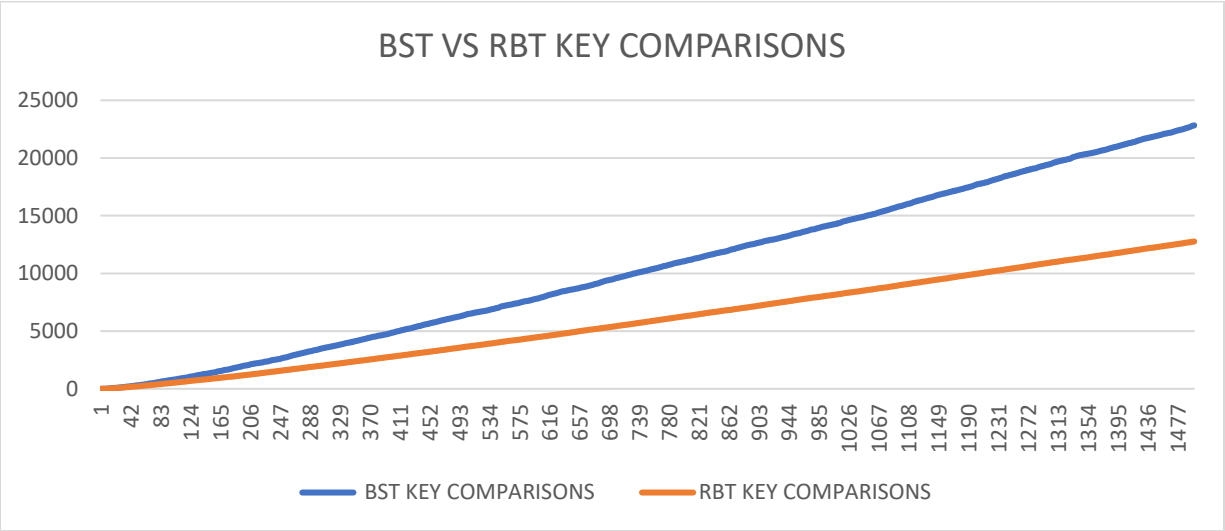
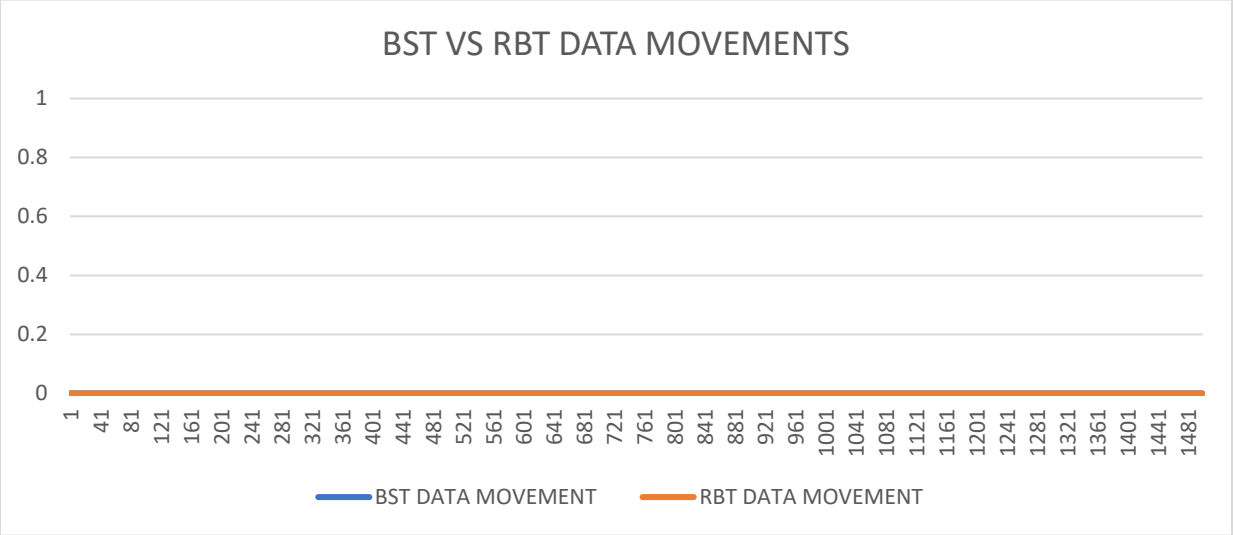
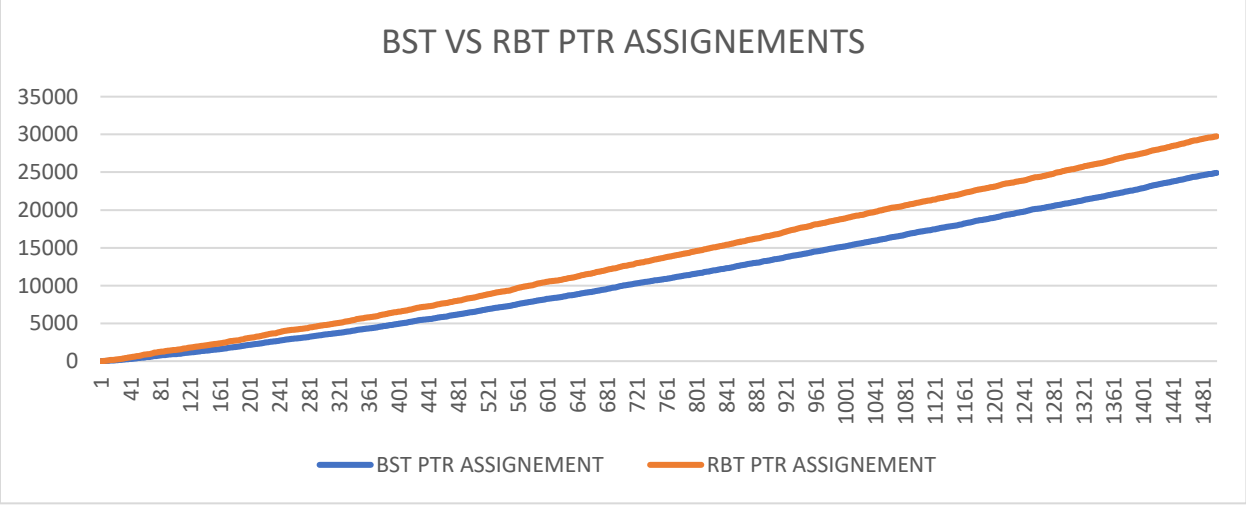
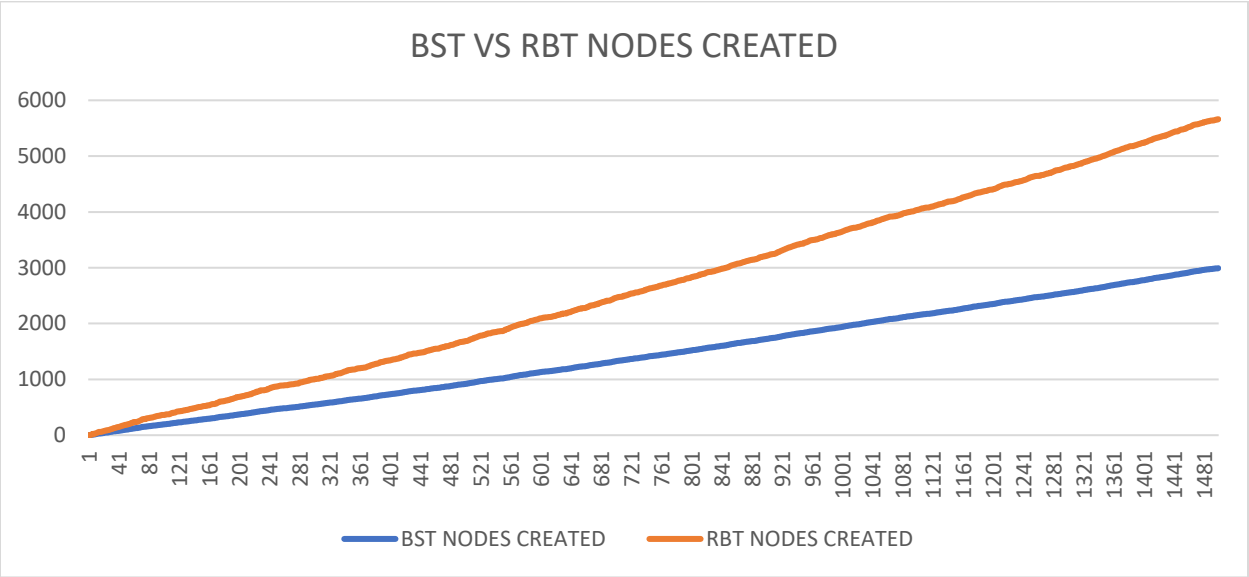
Ayham Makhamra
Project Report

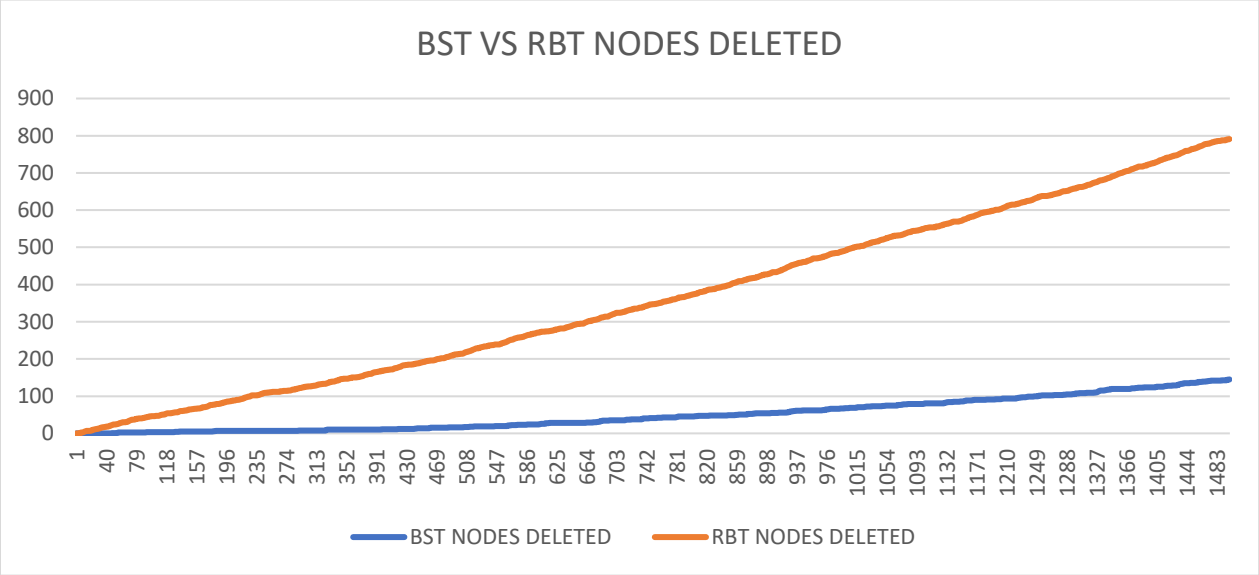






Ayham Makhamra
Project Report





The X-axis for all graph is the number of iterations.

Iteration	BST KEY COMPARISONS	BST DATA MOVEMENT	BST PTR ASSIGNEMENT	BST NODES CREATED	BST NODES DELETED	RBT KEY COMPARISONS	RBT DATA MOVEMENT	RBT PTR ASSIGNEMENT	RBT NODES CREATED	RBT NODES DELETED
1	0	0	0	1	0	0	0	1	1	0
100	781	0	929	193	3	518	0	1481	363	46
200	2020	0	2138	370	7	1198	0	3071	684	86
300	3429	0	3491	547	8	1982	0	4751	1001	125
400	4854	0	4917	731	11	2786	0	6508	1338	168
500	6387	0	6489	919	16	3628	0	8386	1684	214
600	7824	0	8236	1130	24	4478	0	10500	2093	271
700	9453	0	9933	1324	35	5361	0	12491	2454	321
800	11047	0	11564	1517	45	6275	0	14541	2825	372
900	12613	0	13337	1723	54	7176	0	16598	3212	428
1000	14177	0	15167	1939	68	8089	0	18861	3642	491
1100	15882	0	17094	2149	79	9015	0	20986	4031	548
1200	17680	0	18957	2347	92	9957	0	23059	4400	601
1300	19432	0	20886	2553	107	10908	0	25285	4802	659
1400	21105	0	22876	2775	124	11825	0	27494	5237	726
1500	22830	0	24909	2992	145	12766	0	29753	5662	791

- The sets of probabilities were used for the three operations (Insert, Delete, Search) respectively:

  9/20 9/20 2/20 : 45% 45% 10%

- The number of iterations used = 1500.

Interpretation of the result:

Most of the BST operations (e.g., search, insert, delete.. etc) take O(h) time where h is the height of the BST. The cost of these operations may become O(n) for a skewed Binary tree. If we make sure that the height of the tree remains O(log n) after every insertion and deletion, then we can guarantee an upper bound of O(log n) for all these operations. The height of a Red-Black tree is always O(log n) where n is the number of nodes in the tree.

Ayham Makhamra
Project Report

BST vs RBT Key Comparisons

The first graph shows the number of key comparisons for BST and RBT. BST grows faster than RBT. This suggests that while both types of trees have more key comparisons as they grow, BST has more comparisons than RBT for the same number of elements. The less value for RBT is likely due to the tree's self-balancing nature, which keeps operations closer to O(log n) even in the worst-case scenarios, unlike BST which can degrade to O(n).

BST vs RBT Data Movements

Since we are not doing any data movements and we only modify pointers of objects. Data movements=0 for both trees.

BST vs RBT Pointer Assignments

The third graph compares pointer assignments between BST and RBT. Both show a close relationship with the number of iterations, but the RBT has a higher count. This is consistent with expectations since RBTs require additional pointer assignments during rotations and recoloring to maintain the red-black properties after insertions and deletions.

BST vs RBT Nodes Created

The fourth graph shows the nodes created for BST and RBT. RBT has a higher rate of node creation. This could be interpreted as RBT NIL nodes that are created for each time a key-bearing node is inserted. Also, sometimes RBT requires additional nodes creation to maintain balance (insert fix up, delete fix up, rotation, ...etc), although typically the creation of new nodes should be similar for both trees since both create one new node per insertion.

BST vs RBT Nodes Deleted

In the fourth graph, we see the nodes deleted for BST and RBT. RBT has more nodes deleted than BST. This might suggest that RBT are undergoing more deletions to maintain balance, but generally, the number of deletions should be directly related to the number

of deletion operations performed by the user or the program, however, deletion of NIL nodes is counted here, and every time we replace a NIL node with a new key-bearing node, we delete the NIL node.

Conclusions from the Graphs

Key Comparisons: BST seem to perform more key comparisons as they grow compared to RBT, which is expected due to the potential imbalance in BST.

Nodes Created/Deleted: RBT seem to have slightly more in both metrics, which may suggest additional operations to maintain balance.

Pointer Assignments: RBT has more pointer assignments due to the necessary rebalancing after insertions and deletions.